

3. Dealing with Document-Based Knowledge

Varying Degrees of Formalization

Highly formalized knowledge:

- highly expressive representation languages needed
- inferencing capabilities
- + advanced functionalities possible
- requires great effort

Formalized but not generalized knowledge:

- pieces of “knowledge in action” (cases)
- case retrieval based on similarity
- data marts (multi-dimensional data cubes)
- + more easily available
- applying the knowledge may be difficult

Terminology-based (semi-structured text):

- terminologically annotated documents
- content-oriented search
- + most knowledge is in documents – makes it better available
- annotation necessary

Unstructured text:

- take documents as they are
- freetext search
- + easy to realize
- low precision and recall

⇒ only formalize as much as needed

HTML is not meant to represent data:

Organizational Memories

Advanced Course

Date: Friday 12-16

Lecturer: Ulrich Reimer

Readings: F. Lehner: Organizational Memories

```
<h2>Organizational Memories</h2>

<b>Advanced Course</b><br>

<i>Date:</i> Friday 12-16<br>

<i>Lecturer:</i> Ulrich Reimer<br>

<i>Readings:</i> F. Lehner: Organizational Memories
```

Why XML?

- HTML is meant for markup of (hyper)text structure
(but rather used for style setting)
- computers cannot interpret the contents of an HTML document – it's just text
- XML solves this by allowing to introduce content-oriented tags

Previous HTML example in XML:

```
<course type="advanced">  
    <title>Organizational Memories</title>  
    <date>  
        <day>Friday</day>  
        <hours>12-16</hours>  
    </date>  
    <lecturer>Ulrich Reimer</lecturer>  
    <readings>F. Lehner: Organizational Memories</readings>  
</course>
```

missing text fragments are generated according to
tag format definitions

XML

- a meta-language for defining a semantic markup vocabulary
- basic data structure is an ordered tree - each tag corresponds to a labeled node
- is only about data, usage is left open (querying, formatting, etc. are defined elsewhere)
- vocabulary (i.e., tags) can be defined freely (via a DTD)

Definition of an XML vocabulary

- Document Type Declaration (DTD)

A DTD is a grammar specification, e.g.:

```
<!ELEMENT course(title,date,lecturer,readings)>
<!ATTLIST course type (basic|advanced) #REQUIRED>
<!ELEMENT title(#PCDATA)>
<!ELEMENT date(day,hours)>
<!ELEMENT day(#PCDATA)>
<!ELEMENT hours(#PCDATA)>
<!ELEMENT lecturer(#PCDATA)>
<!ELEMENT readings(#PCDATA)>
```

DTDs have very limited expressivity:

- no other relationship between elements than nesting
- no typing of values

→ unsuited for real schema modeling

More recent alternative: XML Schema

- XML Schema is XML itself
- rich set of data types
- mechanism for defining new data types
- richer grammar syntax
- provides a name space mechanism to combine XML documents with overlapping vocabulary

(see: <http://www.w3.org/TR/xmlschema-0/>)

Our example in XML Schema:

```
<complexType name="course">
    <attribute name="type" type="course-type"
        use="optional" default="basic"/>
    <element name="title" type="string"
        maxOccurs="1"/>
    <element name="date" type="date-type"/>
    <element name="lecturer" type="string"/>
    <element name="readings" type="string"/>
</complexType>
```

```
<complexType name="date-type">
    <element name="day" type="weekdays" />
    <element name="hours" type="hoursrange" />
</complexType>

<simpleType name="weekdays">
    <restriction base="string">
        <enumeration value="Monday" />
        <enumeration value="Tuesday" />
        <!-- and so on ... -->
    </restriction>
</simpleType>
```

Analogously for the types "course-type", "hoursrange"

Querying

HTML:

screen scraping: use regularities in layout to extract contents

XML:

interpreting structure: possible because tags represent contents

XML query languages (XQL, XML-QL, UnQL, etc.)

- exploit the basic data model (a labeled tree)
- central concept: path expressions

Output formatting for XML

CSS (Cascading Style Sheets)

- for every tag its output format is defined
- output style is inherited by all subordinate tags
(but can be redefined)

XSL (eXtensible Stylesheet Language)

- a general tree transforming language
- a resulting tree can be an HTML document
- a resulting tree can be a tree consisting of XSL
Formatting Objects

Describing the semantics of data

XML separates document structure from its layout

But: semantics and structure are still interwoven

RDF (Resource Description Framework):

- a means for adding semantics to a document
- no assumption about the document structure made
- is an XML application

RDF data model:

- a *resource* is an entity to talk about
 - resources have URIs
 - RDF definitions are itself resources
- a *property* defines a binary relation between resources and/or atomic values
- a *statement* specifies for a resource a value for a property
 - *resource* has *property* with *value*
 - values can be atomic or resources

```

<RDF xmlns = "http://w3.org/TR/1999/PR-rdf-syntax-19990105#"
      xmlns:DC = "http://purl.org/DC#"
      xmlns:S = "http://www.inf-wiss ... /course-schema/" >

  <Description about = "http://www.inf-wiss ... /om.html" >
    <DC:Title> Course Organizational Memories </DC:Title>
    <DC:Date> 2000-09-01 </DC:Date>
    <DC:Creator> Ulrich Reimer </DC:Creator>
    <DC:Subject> OM, Course Description </DC:Subject>
    <S:description-of> resource="OM-Course" </S:description-of>
  </Description>

  <Description about = "OM-Course">
    <S>Type> advanced </S>Type>
    <S>Title> Organizational Memories </S>Title>
  </Description>
</RDF>
```

```

<RDF xmlns = "http://w3.org/TR/1999/PR-rdf-syntax-19990105#"
      xmlns:DC = "http://purl.org/DC#"
      xmlns:SM = "http://www.swisslife ... /skills/" >

<Description about = "http://www.swisslife ... /~reimer" >
  <DC:Title> Home Page of Ulrich Reimer </DC:Title>
  <DC:Date> 2000-09-01 </DC:Date>
  <DC:Creator> Ulrich Reimer </DC:Creator>
  <SM:owner> resource = "Ulrich-Reimer" </SM:owner>
</Description>

<Description ID = "Ulrich-Reimer" >
  <SM:name> Ulrich Reimer </SM:name>
  <SM:person-id> 87531 </SM:person-id> </Description>

<Description about = "Ulrich-Reimer" >
  <SM:skills-of>
    <Bag> <li SM:expert-level="3"> OM </li>
          <li SM:expert-level="3"> KM </li> </Bag>
  </SM:skills-of>
</Description> </RDF>

```

Ulrich Reimer, Swiss Life

3-20

Statements about Statements (Reification)

...

```

<Description about = "Ulrich-Reimer"
              ID = "ur-skills" >
  <SM:skills-of>
    <Bag> <li SM:expert-level="3"> OM </li>
          <li SM:expert-level="3"> KM </li> </Bag>
  </SM:skills-of>
</Description>

```

```

<Description about = "ur-skills" >
  <reliability> high </reliability>
</Description>

```

...

Ontology Modeling

RDF introduces a data model but no vocabulary

RDF Schema allows to define the vocabulary to use in RDF:

- core classes are
 - rdfs:Resource – everything that is described by an RDF statement is an instance of this class
 - rdf:Property – the class of all properties used to characterize instances of rdfs:Resource
 - rdfs:Class – all concepts (classes) are instances of this

- core properties are
 - rdf:type – instance-of relationship between resources and classes
 - rdfs:subClassOf – subsumption hierarchy between classes
 - rdfs:subPropertyOf – subsumption hierarchy between properties
- core constraints are
 - rdfs:range – restrict the values of a property
 - rdfs:domain – restrict the resources to which a property applies

```

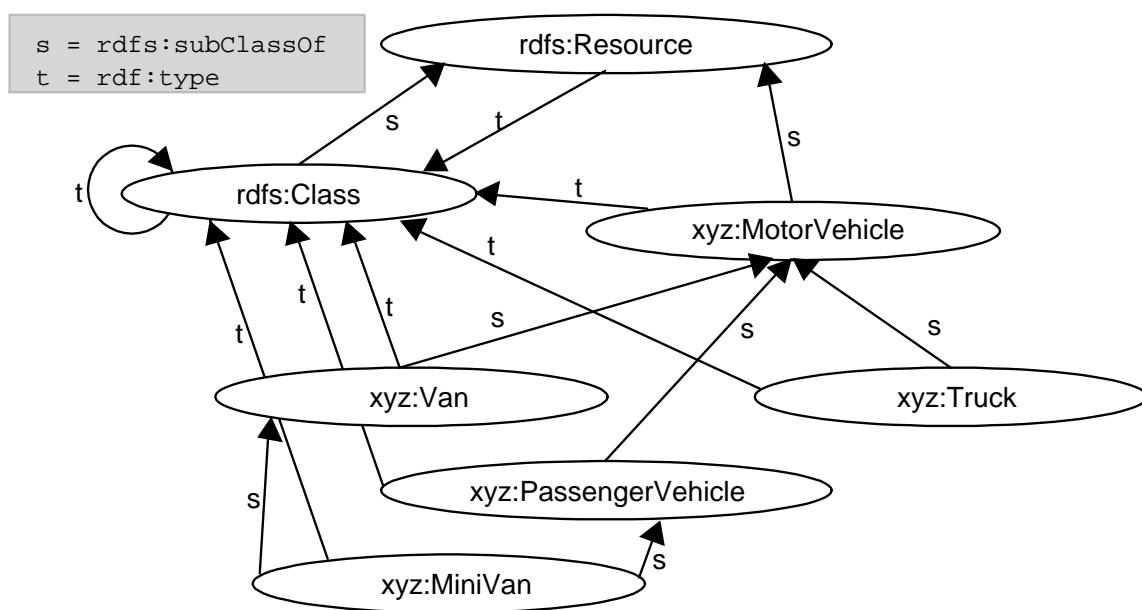
<rdfs:Class rdf:ID="Employee">
  <rdfs:subClassOf="Person" />
</rdfs:Class>

<rdfs:Class rdf:ID="Skills">
  <rdf:type="Class" />
</rdfs:Class>

<rdf:Property rdf:Id="skills-of">
  <rdfs:domain rdf:Resource="Employee" />
  <rdfs:range rdf:Resource="Skills" />
</rdf:Property>

```

Another example:



Querying RDF

SQL/XQL style approach

- views RDF as a relational or XML database
- “select/from/where” style
- path expressions to navigate across properties in an RDF graph
- union, intersection, difference on result sets
- group, order (like in SQL)
- universal and existential quantifiers

(see: www.w3.org/TandS/QL/QL98/pp/rdfquery.html)

Extend RDF Schema into a description logic

- RDF Schema can use its vocabulary to define itself, so can it be used to define other ontology languages
- the description logic OIL is defined in RDF Schema
- expressivity of Core OIL is equivalent to RDF Schema
- extensions of Core OIL as specializations of RDF Schema vocabulary:
 - oil:ClassExpression is a subclass of rdfs:Resource and extends it
 - rdfs:Class is a specific case of a class expression and thus a subclass of oil:ClassExpression
 - slot definitions in OIL are instances of rdf:Property or extend it and are subproperties of rdf:Property

(see: <http://www.ontoknowledge.org/oil/>)